Randomized Permutations in a Coarse Grained Parallel Environment*

[Extended Abstract†]

Jens Gustedt
LORIA & INRIA Lorraine‡

ABSTRACT

We show how to uniformly distribute data at random (not to be confounded with permutation routing) in a coarse grained parallel environment with p processors. In contrast to previously known work, our method is able to fulfill the three goals of uniformity, work-optimality and balance among the processors simultaneously. To guarantee the uniformity we investigate the matrix of communication requests between the processors. We show that its distribution is a generalization of the multivariate hypergeometric distribution and we give algorithms to compute it efficiently.

Categories and Subject Descriptors

D.1.3 [**Programming Techniques**]: Concurrent Programming—*Distributed programming, Parallel programming*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Permutations and combinations*

General Terms

Algorithms, Performance, Theory

Keywords

Randomized permutations, coarse grained parallelism, uniformly generated communication matrix

1. RANDOM PERMUTATIONS

Random permutation of data is a basic step in many computations. It is used e.g (i) to achieve a distribution of the data to avoid load imbalances in parallel and distributed computing, (ii) good generation of random samples to test algorithms and their implementations, (iii) in statistical tests, (iv) in computer games, to only name a few. Creating such permutations is relatively costly: one issue that causes this relatively high cost is the generation of (pseudo-)random numbers, but it is not the only one. Another important issue is that a standard algorithm to generate random permutations addresses memory in an unpredictable way and thus causing a lot of cache misses. The running time of a straight forward permutation program is more or less bound to the cpu-memory latency.

Reducing the cost of such a time consuming subroutine is thus an issue, and here we will present an approach to achieve this. First of all, our approach is a parallel one, i.e uses several processors to compute a random permutation in a parallel or distributed setting. But it also has implications on the sequential framework since allows to relax the constraint imposed by cpu-memory *latency* to one on cpu-memory *bandwidth*.

When designing an alternative to the straight forward random generation of permutations, we have to ensure that we do not loose upon its quality: assuming that we have a "real" generator of random numbers we want each possible permutation to occur equally likely.

2. A COARSE GRAINED FRAMEWORK

Our goal is to describe a realistic framework for the generation of random permutations. As a real suitable family of models of parallel computation we use a coarse grained setting that was derived from BSP, see Valiant [1990], which is called PRO, see Gebremedhin et al. [2002]. PRO allows the design and analysis of scalable and resource-optimal algorithms. It is based on (i) relative optimality compared to a fixed sequential algorithm as an integral part of the model; (ii) measuring the quality of the parallel algorithm according to its granularity, that is the relation between p and n. In particular, coarseness here means $p \leq \sqrt{n}$, for p the number of processors and n the number items. Whereas solving the problem for a broader range of p might be intellectually challenging, it is not of much relevance for real architectures in a foreseeable future.

PRO only considers algorithms that are both time- and space-optimal. As a consequence of this enforced optimality, a PRO-algorithm always yields linear speed-up relative to a reference sequential algorithm. In addition, PRO assumes that the coarse grained communication cost only depends on p and the bandwidth of the considered point-to-point interconnection network. By that it allows for a design and analysis of algorithms that is still relatively simple but at the same time ensures efficiency of implementations on a wide rage of platforms, see Essaïdi et al. [2002].

The use of such models is in contrast to the assumptions that are made by Czumaj et al. [1998] (which solve the

*This work has been supported by the Lorraine region via the programme PRST IL.

†This abstract is a short version of Gustedt [2002]

‡Campus scientifique, BP 239, F-54506 Vandœuvre-lès-Nancy, France; Email: Jens.Gustedt@loria.fr

SPAA'03, June 7–9, 2003, San Diego, California, USA.
ACM 1-58113-661-7/03/0006.

problem by simulating some fine grained sorting network) and algorithms developed in for the PRAM setting (see eg Hagerup [1991]) and which are either not cost-optimal or not balanced between processors. Please also note, that the so-called *permutation routing* problem (see eg Kruskal et al. [1990]) as it was intensively studied for the BSP and similar models is very different from our problem here. There one tries to optimize the communication of the messages during one superstep, the so-called h-relation.

Goodrich [1997] proposed an algorithm for our problem on the BSP which uses general sorting as a subroutine. By that this algorithm has a superlinear total cost ($\log n$ per item) and is not work-optimal. Guérin Lassous and Thierry [2000] investigated several other algorithms to compute random permutations in a coarse grained setting.

3. CONCURRENCE OF THREE GOALS

Up to our knowledge, none of the previous algorithms concurrently fulfilled the following three goals.

Uniformity: provided we have a perfect generator of randomness, all permutations must appear equally likely;

Work-optimality: to be suitable for useful implementations the total work (including communication and generation of random numbers) must at least asymptotically be the same as in a sequential setting;

Balance: during the course of the algorithm none of the processors must be overloaded with work or data.

Especially uniformity and balance seem to work against each other. A typical trick to obtain balance for an algorithm that has some (small) probability of imbalance is to start-over whenever such an imbalance is detected. Usually this works well on the work-optimality when probabilities are small enough, and only increases the average running time a bit. But this also means that certain runs that lead to valid permutations are rejected. It is not even to see that all permutations can be obtained nor is it in general possible to prove uniformity.

Another trick to avoid imbalance and non-uniformity is to iterate. If we have a method that is non-uniform but balanced we can iterate it to obtain a uniform distribution. Usually this needs a logarithmic number of iterations and so the total work is a log-factor away from optimality.

THEOREM 1. *A network of p homogeneous processors may uniformly sample a random permutation of size $n = p \cdot m$, $p \leq m$, such that the usage of the following resources is $O(m)$ per processor and thus $O(n)$ in total: memory, computation time, random numbers and bandwidth.*

4. THE COMMUNICATION MATRIX

We achieve our goal by first computing a matrix $A = (a_{ij})$ that accounts the amount of communication from processor P_i to processor P_j. A is not an arbitrary matrix but has special properties. In particular we have that the row and column sums are all equal to m.

If we suppose we are given such a matrix A, a permutation can be realized by sending out a_{ij} items between all pairs of processors. As we aim for a random permutation we must ensure that (i) the individual items that are send from P_i to P_j are chosen arbitrarily and that (ii) all items that are received on P_j are mixed randomly. This can easily be achieved by two local random permutations, one before the communication and the other thereafter.

So what remains to be done is the efficient sampling of such matrices under the appropriate probability distribution. Since we want a uniform distribution for the permutations not all matrices A occur with the same probability. The distribution is not trivial and in particular the different a_{ij} are not independent.

It turns out that the distribution for these matrices is a generalization of the hypergeometric distribution, see Siegrist [2001] for the related probability theory. In fact, the technical part of the paper shows how for the recursive computation of a split into independent submatrices it suffices to locally compute hypergeometric distribution on individual processors. The computation of that distribution can be easily and efficiently be done by the theory and code as given in Zechner [1994]. By our technique we are able to give an algorithm that samples the matrix with the correct distribution. It has an optimal work load and communication of $O(m)$ per processor and a number of communication rounds (or *supersteps*) of $O(\log p)$, see Gustedt [2002].

References

A. Czumaj, P. Kanarek, M. Kutylowski, and K. Lorys. Fast generation of random permutations via networks simulation. *Algorithmica*, 21(1):2–20, 1998.

Mohamed Essaïdi, Isabelle Guérin Lassous, and Jens Gustedt. SSCRAP: An environment for coarse grained algorithms. In *Fourteenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2002)*, 2002.

Assefaw Hadish Gebremedhin, Isabelle Guérin Lassous, Jens Gustedt, and Jan Arne Telle. PRO: a model for parallel resource-optimal computation. In *16th Annual International Symposium on High Performance Computing Systems and Applications*, pages 106–113. IEEE, The Institute of Electrical and Electronics Engineers, 2002.

Michael T. Goodrich. Randomized fully-scalable BSP techniques for multi-searching and convex hull construction. In Michael Saks et al., editors, *Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–776. SIAM, Society of Industrial and Applied Mathematics, 1997.

Isabelle Guérin Lassous and Éric Thierry. Generating random permutations in the framework of parallel coarse grained models. In *Proceedings of OPODIS'2000*, volume 2 of *Studia Informatica Universalis*, pages 1–16, 2000.

Jens Gustedt. Randomized permutations in a coarse grained parallel environment. Technical Report RR-4639, INRIA, November 2002.

Torben Hagerup. Fast parallel generation of random permutations. In *Automata, Languages and Programming*, volume 510 of *Lecture Notes in Comp. Sci.*, pages 405–416. Springer-Verlag, 1991. Proceedings of the 18th International Colloquium ICALP'91.

Clyde P. Kruskal, Larry Rudolph, and Marc Snir. A complexity theory of efficient parallel algorithms. *Theoretical Computer Science*, 71 (1): 95–132, march 1990.

Kyle Siegrist. *Virtual Laboratories in Probability and Statistics*, chapter C.4, Finite Sampling Models: The Multivariate Hypergeometric Distribution. University of Alabama, 2001. URL `http://www.math.uah.edu/stat/urn/urn4.html`.

Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

H. Zechner. *Efficient sampling from continuous and discrete unimodal distributions.* PhD thesis, Technical University Graz, Austria, 1994.